Research report 155

IMPROVED CIRCUITS AND FORMULAE FOR MULTIPLE ADDITION, MULTIPLICATION AND SYMMETRIC BOOLEAN FUNCTIONS

Michael S Paterson, Uri Zwick

(RR155)

Circuits and formulae over several bases, with improved depth and size respectively, are given for the carry save addition of n binary numbers (i.e., obtaining two numbers whose sum is equal to the sum of the n numbers). As a consequence improved circuits and formulae for multiple addition, multiplication and for most symmetric Boolean functions are obtained.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

March 1990

Improved circuits and formulae for multiple addition, multiplication and symmetric Boolean functions

Michael S. Paterson * *Uri Zwick* †

Preliminary version: March, 1990

Abstract

Circuits and formulae over several bases, with improved depth and size respectively, are given for the carry save addition of n binary numbers (i.e., obtaining two numbers whose sum is equal to the sum of the n numbers). As a consequence improved circuits and formulae for multiple addition, multiplication and for most symmetric Boolean functions are obtained.

1 Introduction

Ofman [15] and Wallace [27] were the first to show that multiplication of two n-bit numbers can be done in depth $O(\log n)$. They both used a component called a *Carry Save Adder (CSA)* which reduces the sum of 3 numbers to the sum of 2 numbers in a (small) constant depth. It is easy to see that using $\log_{3/2} n + O(1)$ levels of *CSA*'s one can reduce the sum of n numbers to the sum of only 2 numbers. Such constructions are usually called *Wallace trees* in the more practical literature (e.g.,

*Department of Computer Science, University of Warwick, Coventry, CV4 7AL, England. This author was partially supported by a Senior Fellowship from the SERC and by the ESPRIT II BRA Programme of the EC under contract # 3075 (ALCOM).

†Department of Computer Science, University of Warwick, Coventry, CV4 7AL, England. This author was partially supported by a Joseph and Joan Nissim postdoctoral grant from Tel-Aviv University.

see [1]). The resulting two numbers can now be added (if required) using additional depth $\log n + O(\sqrt{\log n})$ (see [3], [9]).

In this work we show that the depth of the previous construction can be reduced using the fact that some of the inputs may be supplied to a *CSA* later than the others. For any specific design of a *CSA*, our constructions appear to give asymptotically optimal combinations of such *CSA*'s with respect to depth or formula size for the reduction of n summands. Results for depth similar to ours were recently obtained independently by Nick Pippenger. His results are somewhat more general than ours since he analyzes real-valued delays, not just integer-valued depths as we do.

A slight modification of Wallace trees was given by Dadda [6]. Khrapchenko (see [13]) presented a sub-optimal analysis of *CSA* networks and showed that n summands can be reduced to 2 in depth $5.12 \log n + O(1)$ using U_2-circuits (i.e., circuits utilizing only the unate 2-input binary functions). We improve his results and design depth $5.07 \log n$ U_2-circuits and $3.71 \log n$ B_2-circuits (i.e., circuits utilizing all the 2-input binary functions) for the same problem.

Since carry save adders have size linear in the number of their inputs, the circuits we get for multiplication have $O(n^2)$ size. Multiplication circuits of size $O(n \log n \operatorname{loglog} n)$ were constructed by Schönhage and Strassen [23] (see also [2],[14]). These circuits use the FFT algorithm (see [2]) and also have logarithmic depth. However the constant factor in their depth seems to be much larger than ours. Optimizing with respect to depth thus appears to give rise to different circuits from those obtained when optimizing for size. Our present work is mainly concerned with asymptotic results, but more sensitive implementations of the circuits described here may have practical applications.

Carry save adders can be used not only for multiplication but also for *multiple addition* (i.e., the summation of $n > 2$ numbers) and as a special case for *counting* (i.e., the summation of $n > 2$ bits). The result of a count can be used to compute the *majority* function (i.e., the function which outputs 1 if and only if at least half of the arguments are 1) or indeed any *symmetric Boolean* function (i.e., a function which depend only on the number of inputs which are 1). The same depth results that we stated for multiplication are also obtained for multiple addition, for majority and for a wide family of other symmetric Boolean functions. For general symmetric Boolean functions a slightly larger depth is required.

Note that logarithmic depth circuits imply polynomial size formulae. We were

not able to find any reference to previous work on the formula complexity of multiplication or multiple addition but the formula complexity of the symmetric Boolean functions was extensively studied (see [10],[11],[12],[16],[17],[18],[26]). The best previously known B_2-formulae for majority and for general symmetric Boolean functions were obtained by Peterson (see [17]), who improved on results of Pippenger and Paterson (see [16],[18]) and obtained formulae of size $O(n^{3.32})$ for majority and of size $O(n^{3.37})$ for all symmetric functions. Khrapchenko (see [12]) obtained U_2-formulae of size $O(n^{4.62})$ for the majority function and of size $O(n^{4.93})$ for general symmetric Boolean functions.

In this work we improve the results mentioned above and obtain B_2-formulae of size $O(n^{3.16})$ and of size $O(n^{3.30})$ respectively and U_2-formulae of sizes $O(n^{4.60})$ and $O(n^{4.85})$ respectively for majority and for general symmetric Boolean functions.

The bounds stated for majority also hold, for example, for all the MOD_k functions for fixed k (MOD_k is the function which outputs 1 iff the number of its arguments which are 1 is divisible by k). For some special values of k better formulae are known. If $k = 2^p$ then $O(n(\log n)^{p-1})$ B_2-formulae are easily constructed (see [7]). Van Leijenhorst [26] obtained $O(n^2)$ B_2-formulae for MOD_3, $O(n^{2.58})$ B_2-formulae for MOD_7 and $O(n^3)$ B_2-formulae for MOD_5. Trivial constructions give $O(n^{1+\log k})$ U_2-formulae for MOD_k. These bounds are better than our uniform bound for $k \leq 12$.

The threshold functions $TH_{n,k}$ ($TH_{n,k}$ is the function of n variables which outputs 1 iff at least k of the input variables are 1) can all be obtained as restrictions of majority functions. Our upper bounds apply therefore to them as well. However, for fixed k much better formulae are known. Friedman (see [8]) gave explicit constructions of monotone formulae (i.e., formulae using only dyadic AND and OR) of size $O(k^c n \log n)$ for $TH_{n,k}$ (with some large c). Boppana (see [5]) gave a nonconstructive proof for the existence of monotone formulae of size $O(k^{4.28} n \log n)$ for $TH_{n,k}$. This result with $k = n/2$ (i.e., majority) was first obtained by Valiant (see [25]).

Depth and the logarithm of the formula size are closely connected. It is known for example that $\log L_{B_2}(f) \leq D_{B_2}(f) \leq 2.47 \log L_{B_2}(f)$ (in fact even that $D_{U_2}(f) \leq 2.47 \log L_{B_2}(f)$) and that $\log L_{U_2}(f) \leq D_{U_2}(f) \leq 1.81 \log L_{U_2}(f)$ (see [4],[20],[22]). These inequalities are not tight enough for our purposes and we will optimize differently for depth and for formula size. The known connections between B_2 and U_2, namely $D_{U_2}(f) \leq 2D_{B_2}(f)$ and $L_{U_2}(f) \leq O\left((L_{B_2}(f))^{\log_3 10}\right)$ (see [19]), are also too crude to be of any help to us, and separate optimizations for B_2 and for U_2 will be

made.

The optimization of formula size will be more complicated than that of depth. Some of the parameters to be adjusted are necessarily integral for depth but can be fractional for formula size.

The formulae for carry save addition will again be obtained by composing basic '$3 \rightarrow 2$' or '$7 \rightarrow 3$' carry save adders. It will be however more convenient to describe the formulae as compositions of *Full Adders (FA's)* which are the basic building blocks from which carry save adders are made. Again, our construction appears to use a given *FA* 'technology' optimally.

The gap between the upper bounds obtained here and the lower bounds known is still very big. Over B_2 the largest lower bound on the formula complexity of a symmetric function is an $\Omega(n \log n)$ lower bound obtained by Fischer, Meyer and Paterson (see [7]). This lower bound holds for almost all the symmetric functions including majority and MOD_k for $k > 2$ (it is tight for $k = 4$). Over U_2 Khrapchenko (see [11]) obtained an $\Omega(n^2)$ lower bound which holds for many symmetric functions including MOD_k for $k \geq 2$ (it is tight for $k = 2$) and majority.

The rest of the paper is organized as follows. In the next section we formally define the arithmetical problems involved and obtain some connections between their complexities. In section 3 we present the basic building blocks used in our constructions. These are FA_3's and FA_7's (i.e., *FA*'s which add up 3 or 7 input bits respectively). The FA_3's presented are well known and easily seen to be optimal. A new construction of a FA_7 over B_2 is then presented. These FA_7's are used to construct our best B_2-formulae. For the basis U_2, an FA_7 obtained by Khrapchenko (see [12],[13]) is used for our best U_2-depth.

In section 4 we describe our basic formula construction. For the sake of simplicity this basic construction uses FA_3's and therefore does not yet give our optimal results. In the subsequent section (section 5) we present an abstract continuous model in which the values obtained by our basic construction can be derived in a more straightforward manner. We can use this model to compute the results obtained when using any given *FA* as the basic building block, and in section 6 we derive our best formulae. In section 7 we describe our depth constructions in a similar style.

2 Arithmetical Problems

We begin by formal definitions of multiple addition, carry save addition, multiplication, carry save multiplication, counting, modular functions, threshold functions, majority and the universal symmetric Boolean functions generator.

Notation If $b = (b_0, \ldots, b_{k-1})$ is a binary vector, we denote by $|b|$ the number that b represents in the binary system, i.e., $|b| = \sum_{j=0}^{k-1} b_j 2^j$.

Definitions

$ADD(n,m)$ is the function which receives as input n binary vectors $a_0, \ldots, a_{n-1}$ of length m and outputs the binary vector b of length $m + \lceil \log(n+1) \rceil$ which satisfies $|b| = \sum_{i=0}^{m-1} |a_i|$. We also define $ADD_j(n,m) = b_j$.

$CSA(n,m)$ is any function which receives as input n binary vectors $a_0, \ldots, a_{n-1}$ of length m and outputs two binary vectors of length $m + \lceil \log(n+1) \rceil$ which satisfy $|b_0| + |b_1| = \sum_{i=0}^{m-1} |a_i|$. Notice that $CSA(n,m)$ is a family of functions and when we speak about the complexity of $CSA(n,m)$ we refer to the minimum of the complexities of functions in this family. When we confine our attention to a fixed function in this family we also define $CSA_j(n,m) = (b_{0j}, b_{1j})$.

The operations performed respectively by $ADD(n,m)$ and $CSA(n,m)$ are called *multiple addition* and *(multiple) carry save addition.*

$MUL(n)$, the usual multiplication function, receives as input two binary vectors a, b of length n and returns a third vector c of length $2n$ satisfying $|c| = |a| \cdot |b|$.

$CSM(n)$ (or *carry save multiplication*) is defined similarly to $MUL(n)$ but, as for carry save addition, the result is represented as the sum of two numbers.

$CNT(n)$ is the *counting function* which receives a binary vector x of length n and returns a binary vector y of length $\lceil \log(n+1) \rceil$ satisfying $|y| = \sum_{i=0}^{n-1} x_i$.

$TH(n,k)$, the k^{th} *threshold* function), receives a binary vector of length n and outputs 1 iff $\sum_{i=0}^{n-1} x_i \geq k$. The function $MAJ(n) = TH(n, \lceil n/2 \rceil)$ is called the *majority* function.

$SYM(n)$ is the function which receives a binary vector x of length n and a binary vector a of length $n+1$ and outputs the bit a_k where $k = \sum_{i=0}^{n-1} x_i$. Since any symmetric Boolean function can be obtained as a restriction of $SYM(n)$ by choosing an appropriate vector a, this function is called the *universal symmetric Boolean function generator.*

Some elementary relationships among the formula complexities of these functions can be shown. The formula complexity of a function which returns more than one bit is taken to be the maximum over the formula complexity of the individual bits. For the sake of brevity we denote the formula complexity of a function by the same symbol used to denote the function itself. The connections specified hold for all complete bases. Some multiplicative constant factors may differ from basis to basis. Similar connections may be derived for depth.

Theorem 2.1 *For every complete binary basis* Ω,

1. $ADD(2,m) \leq c_\Omega m + O(1)$, *where* $c_{B_2} = 4$ *and* $c_{U_2} = 8$
2. $ADD(n,m) \leq CSA(n,m) \cdot ADD(2, m + \lceil \log(n+1) \rceil)$
   $= O(CSA(n,m) \cdot (m + \log n))$
3. $CNT(n) = ADD(n,1) = O(CSA(n,1) \cdot \log n)$
4. $CSA(n,m) \leq CNT(n) \cdot CSA(\lceil \log(n+1) \rceil, m + \lceil \log(n+1) \rceil)$
   $= O(CNT(n) \cdot \log^{c_\Omega} n)$ *where* $c_{B_2} \leq 3.16$ *and* $c_{U_2} \leq 4.60$
5. $MUL(n) \leq O(ADD(n,2n))$
6. $CSM(n) \leq O(CSA(n,2n))$
7. $SYM(n) \leq c_\Omega n \sum_{k=0}^{\lceil \log(n+1) \rceil} CNT_k(n)/2^k$ *where* $c_{B_2} = 1$ *and* $c_{U_2} = 2$
8. $MAJ(2^n - 1) = CNT_{n-1}(2^n - 1)$
9. $MOD_k(n) = O(CNT(n) \cdot CSA(\lceil \log(n+1) \rceil, \log k))$
   $= O(CNT(n) \cdot \log^{c_\Omega} n)$ *where* $c_{B_2} \leq 3.16$ *and* $c_{U_2} \leq 4.60$.

In the proof of these inequalities we will use the results that $CSA(n,m) = O(n^{c_\Omega})$ where $c_{B_2} \leq 3.16$ and $c_{U_2} \leq 4.60$ which are proved in section 6. We first comment on some of these relations. *(1)* states that addition has linear formulae. *(2)* makes a connection between (multiple) addition and (multiple) carry save addition. In particular, for numbers of polylogarithmic length (and therefore polynomial size) the formula complexities of the two functions differ by only a polylogarithmic factor. *(3)* and *(4)* provide perhaps the most interesting result. They show that the formula sizes of carry save addition and counting differ by only a polylogarithmic factor.

3 The Building Blocks

A *k-bit full adder* (FA_k) receives a vector of k bits and outputs a vector of $\lceil \log(k+1) \rceil$ bits representing, in binary notation, the number of bits in the input vector which are 1. In fact FA_7 is simply a component which implements the function CNT_k defined in the previous section (and it therefore perhaps should have been call a *basic counter*). Usually k will be of the form $2^\ell - 1$.

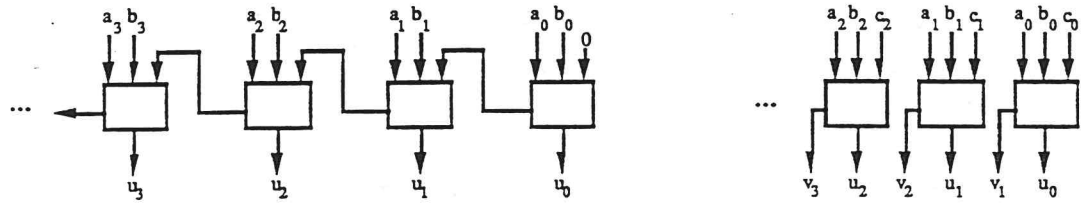

Figure 1. (a) Addition of 2 numbers. (b) Carry save addition of 3 numbers.

Cascades of 3-bit full adders could be used for the addition of 2 numbers, as shown in Figure 1(a), or for the carry save addition of 3 numbers, as shown in Figure 1(b). A carry save adder avoids carry propagation and therefore its depth does not depend on the length of the summands.

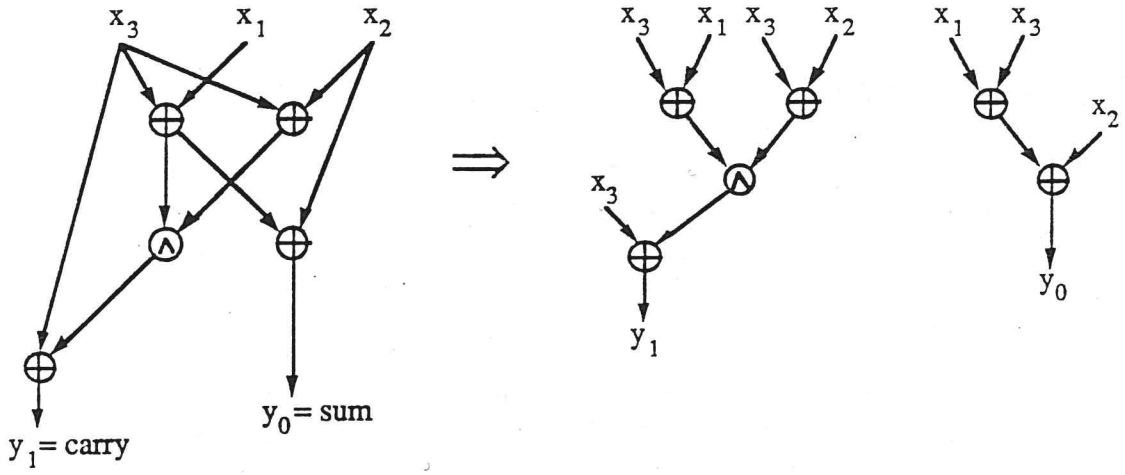

Figure 2. (a) A B_2-FA_3 circuit. (b) Corresponding B_2-FA_3 formulae.

A B_2 implementation of an FA_3 is shown in Figure 2(a). It can be checked that it is minimal with respect to size (in fact Red'kin [21] proved that a cascade of such

FA_3's is a minimal size circuit for addition. For more results on size see [24]). The formulae obtained by expanding this implementation are presented in Figure 2(b).

The formula obtained for y_1 has size 5 (the number of occurrences of variables in it[1]). Each of x_1, x_2 appear in the formula once while x_3 appears 3 times. We therefore say that the *occurrence vector* of the formula is $(1,1,3)$. We also say that the *occurrence matrix* of the FA_3 obtained is $\binom{1\ 1\ 1}{1\ 1\ 3}$.

If $f(x_1,\ldots,x_k)$ is a formula with occurrence vector $(a_1,\ldots,a_k)$ and $g_1,\ldots,g_k$ are formulae of sizes $\ell_1,\ldots,\ell_k$ then the size of the formula $f(g_1,\ldots,g_k)$ is $\sum a_i\ell_i$. If several formulae could be used for f, the one which gives minimal size formula for the composition is not necessarily of minimal size itself.

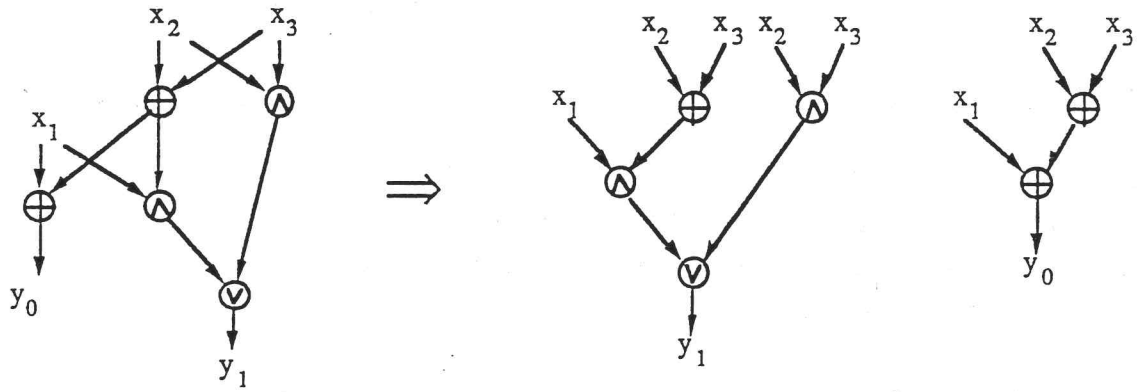

Figure 3. (a) Alternative B_2-FA_3 circuit. (b) Corresponding formulae.

An alternative design of a B_2-FA_3 is given in Figure 3(a). This design is also minimal with respect to size. The formulae obtained by expanding it are shown in Figure 3(b). The formula for y_1 again has size 5 but this time with occurrence vector $(1,2,2)$.

Since $y_1(x_1,x_2,x_3)$ is simply the majority of x_1,x_2,x_3 which is symmetric, we can permute the order of the arguments whenever convenient. It is then easy to see that a $(1,1,3)$ formula for y_1 is always at least as useful as a $(1,2,2)$ formula[2]. However the formula obtained this time has better depth characteristics. The total delay of both the formulae presented for y_1 is 3 but in the new one x_1 could be supplied one unit

[1]In some cases the size of a formula is defined to be the number connectives or gates used in it. If all the connectives are binary then the two definitions differ by exactly 1.

[2]This is because if $\ell_1 \geq \ell_2 \geq \ell_3$ then $\ell_1+\ell_2+3\ell_3 \leq \ell_1+2\ell_2+2\ell_3$.

of time after x_2 and x_3. We say that the first formula has *delay vector* $(3,3,3)$ while the second has delay vector $(2,3,3)$. The formula for y_0 in the second construction has delay vector $(1,2,2)$ and the whole construction has *delay matrix* $\begin{pmatrix} 1 & 2 & 2 \\ 2 & 3 & 3 \end{pmatrix}$.

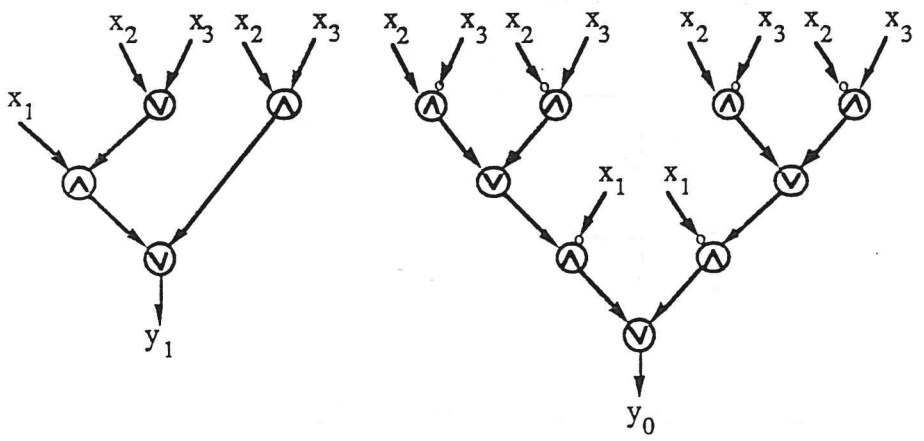

Figure 4. Optimal U_2-FA_3 formulae.

An FA_3 can be implemented using 7 U_2-gates (see [28] for a diagram and for further results on U_2-size). However expanding such an FA_3 gives rise to suboptimal formulae. The optimal U_2-formulae for y_0 and y_1 are given in Figure 4. The resulting U_2-FA_3 has occurrence matrix $\begin{pmatrix} 1 & 2 & 2 \\ 2 & 4 & 4 \end{pmatrix}$ and delay matrix $\begin{pmatrix} 2 & 3 & 3 \\ 2 & 4 & 4 \end{pmatrix}$.

We next move to the construction of a B_2-FA_7. Note that a FA_7 can be constructed from 4 FA_3's as depicted in Figure 5. Denote by $M(x_1,x_2,x_3) = (x_1 \oplus x_3)(x_2 \oplus x_3) \oplus x_3$ the optimal 3-bit majority B_2-formula. We can expand the composition shown in Figure 5 and get the following formulae for an FA_7:

$$
\begin{aligned}
y_0 &= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \\
y_1 &= M(x_1 \oplus x_2 \oplus x_6, x_3 \oplus x_4 \oplus x_7, x_5) \oplus M(x_1,x_2,x_6) \oplus M(x_3,x_4,x_7) \\
y_2 &= M(\ M(x_1 \oplus x_2 \oplus x_6, x_4 \oplus x_5 \oplus x_7, x_3),\ M(x_1,x_2,x_6),\ M(x_4,x_5,x_7)\ )
\end{aligned}
$$

A different ordering of the variables was used for y_1 and for y_2. The resulting formulae have the occurrence matrix $\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 3 & 4 & 4 \\ 2 & 2 & 3 & 4 & 4 & 4 & 10 \end{pmatrix}$. We clearly need to do better than a simple composition of FA_3's for any improvement with FA_7's. A slight improvement

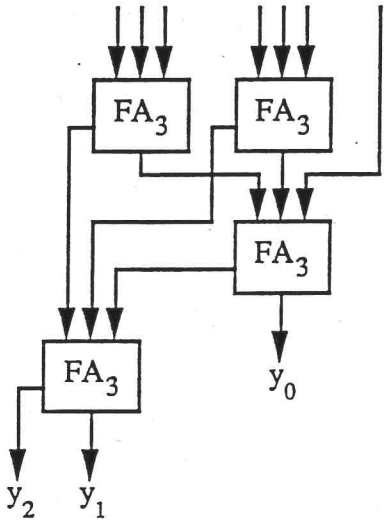

Figure 5. FA_7 as a composition of FA_3's

can be obtained by noticing that

$$\begin{aligned} & M(x_1 \oplus x_2 \oplus x_6, x_3 \oplus x_4 \oplus x_7, x_5) \oplus M(x_1, x_2, x_6) \\ = \quad & (x_1 \oplus x_2 \oplus x_5 \oplus x_6)(x_3 \oplus x_4 \oplus x_5 \oplus x_7) \oplus (x_1 \oplus x_6)(x_2 \oplus x_6) \oplus x_5 \oplus x_6 \\ = \quad & (x_1 \oplus x_2 \oplus x_5 \oplus x_6)(x_3 \oplus x_4 \oplus x_5 \oplus x_7 \oplus 1) \oplus (x_1 \oplus x_6 \oplus 1)(x_2 \oplus x_6 \oplus 1) \oplus 1 \end{aligned}$$

Two literals were eliminated in the last line. Since the above formula appears as a subformula in the expanded formula for y_1 and a similar subformula appears also in the formula for y_2, we get an improved FA_3 with occurrence matrix $\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 3 & 4 \\ 2 & 2 & 2 & 4 & 4 & 4 & 9 \end{pmatrix}$.

It can be easily checked that if y_i is the i^{th} digit in the binary representation of $\sum_j x_j$ then y_i is actually the sum (mod 2) of all the x-monoms of size 2^i (i.e., $y_i = \bigoplus_{\substack{A \subseteq [n] \\ |A|=2^i}} \prod_{j \in A} x_j$). Every y_i is thus an *elementary* symmetric Boolean function. We were not able to use this identity in order to obtain better FA's.

Several constructions of U_2-FA_7's were described by Khrapchenko in [12],[13]. We use his final construction from [13], which has occurrence matrix $\begin{pmatrix} 4 & 8 & 8 & 8 & 8 & 8 & 8 \\ 6 & 8 & 8 & 12 & 12 & 12 & 12 \\ 3 & 4 & 4 & 6 & 6 & 6 & 6 \end{pmatrix}$ and delay matrix $\begin{pmatrix} 4 & 6 & 6 & 6 & 6 & 6 & 6 \\ 5 & 6 & 6 & 7 & 7 & 7 & 7 \\ 6 & 6 & 6 & 8 & 8 & 8 & 8 \end{pmatrix}$.

4 The Basic Construction

Our formulae are formed by the repeated composition of basic formulae which add binary digits. For example, in our basic construction described in this section we make repeated use of the FA_3, which takes as inputs 3 binary digits and outputs 2 digits representing the sum of the three inputs. In our formulae each such input or output digit is naturally associated with a position in the binary representation of the integers. Such a digit is said to have index i if it represents a coefficient of 2^i, $i \geq 0$. Each FA_3 in our formulae, for some i, takes three inputs of index i and outputs one digit of index i and one of index $i+1$. We also classify subformulae roughly according to their sizes. In class $C_{i,j}$, for $i \geq 0$, $j \geq 0$, we will have formulae of index i and of size at most γ^j, where $\gamma > 1$ is to be chosen later. The constant formula 0 will also be considered a member of each class. Note that over the bases considered the input of a constant to a nontrivial formula adds nothing to the resulting size.

Our first construction will use the B_2-FA_3 shown in Figure 2 as its basic building block. If its inputs x_1, x_2, x_3 have sizes s_1, s_2, s_3 respectively the sum and carry digits will have sizes $s_1 + s_2 + s_3$ and $s_1 + s_2 + 3s_3$ respectively. Each FA_3 will be provided with its first two inputs from $C_{i,j+a}$ and its third input from $C_{i,j}$ for some i and j. The sum output will be assigned to class $C_{i,j+b}$ and the carry output to $C_{i+1,j+c}$, where a, b, c are fixed integers satisfying

$$0 \leq a < b < c.$$

We choose $\gamma > 1$ such that

$$1 + 2\gamma^a \leq \gamma^b \quad \text{and} \quad 3 + 2\gamma^a \leq \gamma^c.$$

These inequalities ensure that the sizes of the outputs are appropriate to their classification. We shall say that such an FA_3 is of type $\langle i, j \rangle$.

A construction is fully characterised by specifying for every i and j the number $A_{i,j}$ of FA_3's of type $\langle i, j \rangle$ used in it. It can then be easily seen that the number of formulae of class $C_{i,j}$ generated during the intermediate phases of this construction is

$$Gen_{i,j} \equiv A_{i-1,j-c} + A_{i,j-b}$$

and that the number of class $C_{i,j}$ formulae consumed in this construction is

$$Con_{i,j} \equiv A_{i,j} + 2A_{i,j-a}.$$

If for some i, j we have $Con_{i,j} > Gen_{i,j}$ then, the construction can accept as external inputs $(Con_{i,j} - Gen_{i,j})$ formulae of class $C_{i,j}$. If, on the other hand, $Gen_{i,j} > Con_{i,j}$ then the construction outputs $(Gen_{i,j} - Con_{i,j})$ formulae of class $C_{i,j}$.

We want a construction which will accept N inputs of index i and size 1 for $0 \leq i < m$ and which will output only a finite number of formulae for every given index. Such a construction yields formulae for the carry save addition of N numbers of length m. The construction may well output more than two formulae for each index but the finite number of numbers output could then be carry-save-added using the usual methods contributing only a multiplicative factor to the formula size (or an additive constant to the depth).

In our basic construction we define

$$\mathring{A}_{i,j} = \begin{cases} \lceil K\lambda^i\delta^{-j} + 1 \rceil & \text{if } i \geq 0,\ j \geq 0, \text{ and } K\lambda^i\delta^{-j} \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

where $K, \delta, \lambda \geq 1$ are real parameters to be chosen later. Notice that $A_{i,j} > 0$ only for $0 \leq j \leq L_i$ where $L_i = \lfloor \log_\delta K\lambda^i \rfloor$, that $Gen_{i,j} > 0$ only for $b \leq j \leq L_i + c$ and that $Con_{i,j} > 0$ only for $0 \leq j \leq L_i + a$.

We shall now check that $Gen_{i,j} \leq Con_{i,j}$ for $i \geq 0$ and $0 \leq j \leq L_i$, so that each formula generated in these classes is used as input to a further formula. If $j < b$ then $Gen_{i,j} = 0$, so the condition is satisfied. Otherwise, we have

$$Gen_{i,j} \leq \left\lceil K\lambda^{i-1}\delta^{-j+c} + 1 \right\rceil + \left\lceil K\lambda^i\delta^{-j+b} + 1 \right\rceil < K\lambda^i\delta^{-j}\left(\frac{\delta^c}{\lambda} + \delta^b\right) + 4,$$

$$Con_{i,j} = \left\lceil K\lambda^i\delta^{-j} + 1 \right\rceil + 2\left\lceil K\lambda^i\delta^{-j+a} + 1 \right\rceil \geq K\lambda^i\delta^{-j}(1 + 2\delta^a) + 3.$$

A choice of δ and λ such that

$$\delta^c/\lambda + \delta^b \leq 1 + 2\delta^a$$

ensures that $Gen_{i,j} < Con_{i,j} + 1$, and therefore that $Gen_{i,j} \leq Con_{i,j}$.

For $0 \leq j < b$, we have $Gen_{i,j} = 0$ and so $\sum_{j=0}^{b-1} Con_{i,j}$ input bits of size 1 can be introduced. We use no other input bits and introduce 0's wherever necessary. Assuming $\lambda \geq 1$, we have $Con_{i,0} \geq K$ and we can choose $K = N$ in this construction.

The output formulae of the construction are of class $C_{i,j}$ for $i \geq 0$ and $L_i < j \leq L_i + c$. The number of output formulae of type $C_{i,j}$ is bounded above by

$$Gen_{i,j} \leq Gen_{i,L_i+1} \leq \left\lceil K\lambda^i\delta^{-L_i+c-1} + 1 \right\rceil + \left\lceil K\lambda^i\delta^{-L_i+b-1} + 1 \right\rceil < (\delta^c + 1) + (\delta^b + 1).$$

In the last inequality we used the fact that $K\lambda^i\delta^{-L_i-1} < 1$ (by the definition of L_i). We therefore get only a finite number of output formulae for every given index. Notice that the size of each one of the output formulae of index i is bounded by

$$\gamma^{L_i+c} = O(\gamma^{L_i}) = O((N\lambda^i)^{\log_\delta \gamma}).$$

We have thus proved the following theorem :

Theorem 4.1 *Let $0 \leq a < b < c$ be positive integers. If $\gamma, \delta, \lambda \geq 1$ satisfy the following conditions :*

$$1 + 2\gamma^a \leq \gamma^b,$$

$$3 + 2\gamma^a \leq \gamma^c,$$

$$1 + 2\delta^a \geq \delta^c/\lambda + \delta^b,$$

then there exist families of B_2-formulae for the carry save addition of N numbers of arbitrary length such that the size of the formulae for the index i bits of the result (i.e., $CSA_i(n, \infty)$) is $O((N\lambda^i)^{\log_\delta \gamma})$.

By choosing appropriate sets of parameters we can also get :

Theorem 4.2 *If α, β are real numbers satisfying the inequality $1+2\alpha^\beta > (2\alpha+1)^\beta + \lambda^{-1}(2\alpha+3)^\beta$ then there exist families of B_2-formulae for the carry save addition of N numbers of arbitrary length such that the size of the (two) formulae for the index-i output bits is $O((N\lambda^i)^{1/\beta})$.*

Proof : For every a define $b = \lceil a \log_\alpha(2\alpha+1) \rceil$ and $c = \lceil a \log_\alpha(2\alpha+3) \rceil$. Also define $\gamma = \alpha^{1/a}$ and $\delta = \gamma^\beta$. It is easy to see that

$$\gamma^b \geq \gamma^{a \log_\alpha(2\alpha+1)} = 2\alpha + 1 = 1 + 2\gamma^a$$

$$\gamma^c \geq \gamma^{a \log_\alpha(2\alpha+3)} = 2\alpha + 3 = 3 + 2\gamma^a.$$

As $a \to \infty$, we in fact have that

$$\gamma^a = \alpha, \quad \gamma^b \to 2\alpha + 1, \quad \gamma^c \to 2\alpha + 3.$$

and therefore also that

$$\delta^a = (\gamma^a)^\beta = \alpha^\beta, \quad \delta^b = (\gamma^b)^\beta \to (2\alpha+1)^\beta, \quad \delta^c = (\gamma^c)^\beta \to (2\alpha+3)^\beta,$$

and therefore when a is big enough we also get that

$$1 + 2\delta^a \geq \delta^b/\lambda + \delta^c.$$

Finally, notice that $\log_\delta \gamma = 1/\beta$ and this finishes the proof.

□

We are interested in the maximal β for which there exists an α such that $1+2\alpha^\beta = (2\alpha+1)^\beta + (2\alpha+3)^\beta$. Notice that for every $\alpha > 0$ there exists a unique β for which this equation is satisfied. Therefore the equation defines β as an implicit function of α. Differentiating this equation with respect to α (treating β as a function of α) and equating to 0 we get a second equation which α and β should satisfy : $\alpha^{\beta-1} = (2\alpha+1)^{\beta-1} + (2\alpha+3)^{\beta-1}$. The optimal values of α and β form the (unique) solution of this transcendental system of two equations in two unknowns. We therefore get:

Corollary 4.1 *Let α, β satisfy the following system of equations:*

$$1 + 2\alpha^\beta = (2\alpha+1)^\beta + (2\alpha+3)^\beta$$
$$\alpha^{\beta-1} = (2\alpha+1)^{\beta-1} + (2\alpha+3)^{\beta-1}.$$

If $\beta' < \beta$ then there exist B_2-formulae of size $O(N^{1/\beta'})$ for the carry save addition of N numbers of arbitrary size.

It can be checked that $\alpha \simeq 2.546905$, $\beta \simeq 0.3119376$ is a solution of the system. Hence,

Corollary 4.2 *There exist B_2-formulae of size $O(N^{3.21})$ for the carry save addition of N numbers of arbitrary size.*

It can be verified that for the output bits with large enough index, the choice $\lambda = 1$ is optimal. This is indeed the case for index $i > 0.92 \log N$. Better bounds for the less significant positions are obtained by choosing $\lambda > 1$. To compute all symmetric Boolean functions we can use families of formulae whose size roughly doubles in passing from one index to the next. This corresponds to choosing $\lambda = 2^\beta$. We find that the optimal parameters in this case are $\alpha \simeq 2.7335445$ and $\beta \simeq 0.4274636$. We then get

Corollary 4.3 *Every symmetric Boolean function of n arguments has a B_2-formula of size $O(n^{3.34})$.*

5 A continuous model

In this section we develop a continuous model in order to simplify the analysis of our formula constructions. This will be needed in the next section when we use FA_7's as the basic building blocks.

In the proof of Theorem 4.2, as a approaches ∞ we have γ approaching 1, i.e., the sizes of adjacent classes become ever closer. In our continuous model we allow 'formulae' of arbitrary real-valued sizes and the discrete formula sizes $\{\gamma^j\}$ defining index classes are replaced by a continuous variable s. For each i the specification of the number of formulae in class $C_{i,j}$ is replaced by a continuous *density function* $f_i(s)$ such that the number of formulae with sizes in the range $[s, se^{\Delta})$ is approximated by $f_i(s)\cdot\Delta$ for small Δ. The integer values $A_{i,j}$ from the last section would be represented by the density functions $h_i(s)$ given by:

$$h_i(s) = \begin{cases} K\lambda^i s^{-\beta} & \text{for } s \geq 1 \text{ and } K\lambda^i s^{-\beta} \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

where $K, \lambda \geq 1, \beta < 1$ are real parameters to be chosen later. Note the correspondence $s^{-\beta} \leftrightarrow \gamma^{-j\beta} = \delta^{-j}$.

Our constructions are designed to make best use of the given basic FA_3 by maintaining the optimal ratios among the sizes of the formulae input to each FA_3. We also have to achieve the balance between incoming and outgoing formulae which was discussed in the previous section. We will introduce the analysis of the continuous model by redoing the construction of Theorem 4.2.

Each FA_3 will be fed by three formulae of sizes $\alpha_1 s, \alpha_2 s, \alpha_3 s$ where $\alpha_1, \alpha_2, \alpha_3$ are real parameters. Without loss of generality we can choose $\alpha_1 = 1$ and $\alpha_t \leq 1$ for all t. We will call such an FA_3 a *type s* FA_3. Assuming that the FA_3 we are using has occurrence matrix $\begin{pmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{pmatrix}$, where $u_t, v_t \geq 1$ for all t, each type s FA_3 will output one index i formula of size $(\alpha_1 u_1 + \alpha_2 u_2 + \alpha_3 u_3)s$ and one index-$(i+1)$ formula of size $(\alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3)s$. The density function for index-i formulae absorbed as inputs by all the FA_3's is $h_i(s/\alpha_1) + h_i(s/\alpha_2) + h_i(s/\alpha_3)$. The density function for index-i

formulae which are output by the FA_3's is $h_i(s/(\alpha_1 u_1+\alpha_2 u_2+\alpha_3 u_3))+h_{i-1}(s/(\alpha_1 v_1+\alpha_2 v_2+\alpha_3 v_3))$. We require that

$$h_i\left(\frac{s}{\alpha_1}\right)+h_i\left(\frac{s}{\alpha_2}\right)+h_i\left(\frac{s}{\alpha_3}\right)\geq h_i\left(\frac{s}{\sum \alpha_t u_t}\right)+h_{i-1}\left(\frac{s}{\sum \alpha_t v_t}\right).$$

The above inequality will be referred to as the *balance condition.* Substituting $h_i(s)=K\lambda^i s^{-\beta}$ in the balance condition for $s\geq \max_t \alpha_t$ and $s^\beta \leq K\lambda^i$, we get that

$$\alpha_1{}^\beta+\alpha_2{}^\beta+\alpha_3{}^\beta \geq (\alpha_1 u_1+\alpha_2 u_2+\alpha_3 u_3)^\beta+\lambda^{-1}(\alpha_1 v_1+\alpha_2 v_2+\alpha_3 v_3)^\beta.$$

If we substitute the appropriate values for u, v, and this time set $\alpha_3=1$ and $\alpha_1=\alpha_2$ (since $u_1=u_2$ and $v_1=v_2$), we obtain the constraint proposed in Theorem 4.2. The strict inequality of Theorem 4.2 is replaced here by a weak inequality.

The carry save addition of N numbers is achieved in an analogous way to the previous section by inserting N input bits into the type-s FA_3's for $1\leq s\leq \min\{\sum \alpha_t v_t, \sum \alpha_t u_t\}$. Since $\min\{\sum \alpha_t v_t, \sum \alpha_t u_t\}\geq 3$, it suffices to choose K such that

$$\int_{s=1}^{3} h_i(s)d(\ln s)\geq N,$$

so the minimal suitable choice satisfies $K=O(N)$.

A similar analysis in the region where $s\simeq h_i^{-1}(1)$ shows that there is a bounded number of output formulae for the index-i bits of the result of size $O(h_i{}^{-1}(1))=O((N\lambda^i)^{1/\beta})$.

For every λ we choose $\alpha_1=1, \alpha_2, \alpha_3$ to maximize $\beta=\beta(\lambda)$, and then for every i we can choose λ to minimize $(N\lambda^i))^{1/\beta(\lambda)}$.

The model easily generalizes for the use of FA_7's. Each type s FA_7 will be fed by seven index-i formulae of sizes $\alpha_1 s, \alpha_2 s, \ldots, \alpha_7 s$ (where again $\alpha_1=1$). At each stage we feed the index-i formulae into FA_7's and get some index-i and index-$(i+1)$ formulae and this time also some index-$(i+2)$ formulae.

Assuming the FA_7 has occurrence matrix $\begin{pmatrix} u_1 & u_2 & \ldots & u_7 \\ v_1 & v_2 & \ldots & v_7 \\ w_1 & w_2 & \ldots & w_7 \end{pmatrix}$, each type-$s$ FA_7 will output one index-i formula of size $(\sum \alpha_t u_t)s$, one index-$(i+1)$ formula of size $(\sum \alpha_t v_t)s$ and one index-$(i+2)$ formula of size $(\sum \alpha_t w_t)s$. All the summations from here on are over $1<t<7$.

Assuming that the density function for FA_7's is $h_i(s)$, the balance condition is

$$\sum h_i\left(\frac{s}{\alpha_i}\right) \geq h_i\left(\frac{s}{\sum \alpha_t u_t}\right) + h_{i-1}\left(\frac{s}{\sum \alpha_t v_t}\right) + h_{i-2}\left(\frac{s}{\sum \alpha_t w_t}\right).$$

Choosing again $h_i(s) = K\lambda^i s^{-\beta}$ we get that

$$\sum {\alpha_t}^\beta \geq \left(\sum \alpha_t u_t\right)^\beta + \lambda^{-1}\left(\sum \alpha_t v_t\right)^\beta + \lambda^{-2}\left(\sum \alpha_t w_t\right)^\beta.$$

We claim again that the size of the formulae obtained for the index i bits of the carry save addition is $O(h_i^{-1}(1)) = O\left((N\lambda^i)^{1/\beta}\right)$. For every λ we can choose $\alpha_1 = 1, \alpha_2, \ldots, \alpha_7$ to maximize β.

Using similar constructions to the one used in section 4 we again get a sequence of constructions which tends to the values given by the continuous model for this FA_7. In particular, replacing the weak inequality in the balance condition by a strict inequality we can get, as we did for the FA_3's, the following result.

Theorem 5.1 *If there exists an FA_7 with occurrence matrix* $\begin{pmatrix} u_1 & u_2 & \ldots & u_7 \\ v_1 & v_2 & \ldots & v_7 \\ w_1 & w_2 & \ldots & w_7 \end{pmatrix}$, *and $\alpha_1, \alpha_2, \ldots, \alpha_7$ and β are real numbers satisfying the inequality*

$$\sum {\alpha_t}^\beta > \left(\sum \alpha_t u_t\right)^\beta + \lambda^{-1}\left(\sum \alpha_t v_t\right)^\beta + \lambda^{-2}\left(\sum \alpha_t w_t\right)^\beta,$$

then there are families of formulae for the carry save addition of N numbers in which the (two) formulae for the index-i output bits are of size $O\left((N\lambda^i)^{1/\beta}\right)$.

It should now be obvious how to generalize the results of this section for larger FA's.

6 Formula size results

After the work done in the previous two sections, all we have to do now is to perform the required numerical optimizations. This can be done using standard numerical analysis methods. The results obtained are presented in the following tables and figures. (These will appear in the final version of the paper.)

In Table 1 the exponents of the formulae obtained for carry save addition using different FA's as building blocks are presented. In order to obtain the exponent of

the formulae obtained for the i^{th} bits of the results one should look for the result corresponding to $\mu = i/\log N$. The last row in the table gives the exponents we get for all the symmetric Boolean functions.

The results presented in Table 1 are also shown in a graphical form in Figure 6 (though more values of μ were used in order to draw these graphs). It can be easily seen that the exponents we obtain for all the symmetric Boolean functions can be "read" from the graph by finding the values of μ at which the tangents to the graphs have a slope of exactly 1, and then adding 1 to the exponents. The corresponding values of μ could be referred to as *Bottle-neck* values. The bottle-neck values for the different *FA* "technologies" are shown in Table 2. Also given there are the values of μ at which the flat parts of the graphs begin.

The main results are also summarized in the following theorem :

Theorem 6.1

(i) There exist B_2-formulae of size $O(n^{3.16})$ for the carry save addition of n numbers.

(ii) There exist B_2-formulae of size $O(n^{3.30})$ for all the symmetric Boolean functions of n variables.

(iii) There exist U_2-formulae of size $O(n^{4.60})$ for the carry save addition of n numbers.

(iv) There exist U_2-formulae of size $O(n^{4.85})$ for all the symmetric Boolean functions of n variables.

7 Depth Constructions

Our construction will again be given in an abstract model using real-valued populations, but now with discrete depths. We begin by describing shallow carry save adders obtained by composing B_2-FA_3's. This time we use the alternative B_2-FA_3 shown in Figure 3. In order to make best use of such *FA*'s we should keep a relative delay of 1 between two of the inputs and the third.

Suppose that, for $i \geq 0$, we use $h_i(d)$ FA_3's which receive index-i bits with depth d, d and $d+1$ and yield one index-i output bit of depth $d+2$ and one index-$(i+1)$ output bit of depth $d+3$. The balance condition this time is

$$2h_i(d) + h_i(d-1) \geq h_i(d-2) + h_{i-1}(d-3).$$

We choose $h_i(d) = K\lambda^i\delta^{-d}$ where K is a constant. Substitution in the balance condition yields

$$\lambda \geq \frac{\delta^3}{2+\delta-\delta^2}.$$

We claim that the depth of the (two) index-i output bits obtained is (up to some additive constant) $f_i^{-1}(0) = \log_\delta N + i\log_\delta \lambda$. The proof is analogous to that for formula size.

Choosing $\lambda = 1$ we find that the carry save addition of N numbers can be done in depth $\log_\delta N \simeq 3.707621 \log N$ where $\delta > 1$ is a root of the polynomial $\delta^3+\delta^2-\delta-2 = 0$. It should now be obvious to the reader how to implement these circuits. We therefore get

Theorem 7.1 *There exist B_2-circuits of depth $\log_\delta N + O(1) \simeq 3.71 \log N + O(1)$, where $\delta > 1$ is a root of the polynomial $\delta^3+\delta^2-\delta-2 = 0$, for the carry save addition of N numbers of arbitrary length.*

References

[1] El Gamal A., Gluss D., Ang P-H., Greene J., Reyneri J., *A CMOS 32 bit Wallace tree multiplier-accumulator. 1986 ISSCC Digest of Technical Papers, pp. 194-195.*

[2] Aho A.V., Hopcroft J.E., Ullman J.D., *The design and analysis of computer algorithms. Addison-Wesley, 1974.*

[3] Brent R., *On the addition of binary numbers. IEEE Trans. on Comp., Vol. C-19 (1970), pp. 758-759.*

[4] Brent R., Kuck D.J., Maruyama K., *The parallel evaluation of arithmetic expressions without division. IEEE Trans. on Comp., Vol. C-22 (1973), pp. 532-534.*

[5] Boppana R.B., *Amplification of probabilistic Boolean formulas. Advances in Computer Research, Vol. 5 on Randomness and Computation, Editor S. Micali, JAI Press, to appear.*

[6] Dadda L., *On parallel digital multipliers. Alta Frequenza, Vol. 45 (1976), pp. 574-580.*

[7] Fischer M.J., Meyer A.R., Paterson M.S., $\Omega(n \log n)$ *lower bounds on length of Boolean formulas. SIAM J. Comput., Vol. 11 (1982), pp. 416-427.*

[8] Friedman J. *Constructing* $O(n \log n)$ *size monotone formulae for the k-th threshold function on n Boolean variables. SIAM J. Comput., Vol. 15 (1986), pp. 641-654.*

[9] Khrapchenko V.M., *Asymptotic estimation of addition time of a parallel adder. Problemy Kibernet., Vol. 19 (1967), pp. 107-122* (in Russian). English translation in *Syst. Theory Res., Vol. 19 (1970), pp. 105-122.*

[10] Khrapchenko V.M., *Complexity of the realization of a linear function in the class of* π*-circuits. Mat. Zametki, Vol. 9 (1971), pp. 35-40* (in Russian). English translation in *Math. Notes Acad. Sciences USSR, Vol. 9 (1971), pp. 21-23.*

[11] Khrapchenko V.M., *Methods of determining lower bounds for the complexity of* π*-schemes. Mat. Zametki, Vol. 10 (1972), pp. 83-92* (in Russian). *Math. Notes Acad. Sciences USSR, Vol. 10 (1972), pp. 474-479* (English translation).

[12] Khrapchenko V.M., *The complexity of the realization of symmetrical functions by formulae. Mat. Zametki Vol. 11 (1972) pp. 109-120* (in Russian). English translation in *Mathematical Notes of the Academy of Sciences of the USSR Vol. 11 (1972) pp. 70-76.*

[13] Khrapchenko V.M., *Some Bounds for the Time of Multiplication. Problemy Kibernet., Vol. 33 (1978), pp. 221-227* (in Russian).

[14] Knuth D.E., *The art of computer programming, Vol. 1 (second edition). Addison-Wesley.*

[15] Ofman Y., *On the algorithmic complexity of discrete functions. Doklady Akademii Nauk SSSR, 145 pp. 48-51* (in Russian). English translation in *Sov. Phys. Doklady, Vol. 7 (1963) pp. 589-591.*

[16] Paterson M.S., *New bounds on formula size. Proc. of 3rd GI conference on Theoretical Computer Science 1977, Lecture Notes in Computer Science 48, Springer-Verlag 1977, pp. 17-26.*

[17] Peterson G.L., *An Upper Bound on the size of formulae for symmetric Boolean functions. Technical Report No. 78-03-01, University of Washington.*

[18] Pippenger N., *Short formulae for symmetric functions. IBM report RC-5143, Yorktown Heights, NY (November 20, 1974).*

[19] Pratt V.R., *The effect of basis on size of Boolean expressions. Proc. 16th IEEE Symposium on FOCS (1975), pp. 119-121.*

[20] Preparata F.P., Muller D.E., *Efficient parallel evaluation of Boolean expressions. IEEE Trans. Comp., Vol. C-25 (1976), pp. 548-549.*

[21] Red'kin N.P., *Minimal realizations of a binary adder.Problemy Kibernet., Vol. 38 (1981), pp. 181-216,272* (in Russian).

[22] Spira P.M., *On time-hardware complexity tradeoffs for Boolean functions. Proc. 4th Hawaii Int. Symp. on System Sciences (1971), pp. 525-527.*

[23] Schönhage A., Strassen V., *Schnelle Multiplikation grosser Zahlen. Computing, Vol. 7 (1971), pp. 281-292.*

[24] Stockmeyer L., *On the combinational complexity of certain symmetric Boolean functions. Math. Syst. Theory, Vol. 10 (1977), pp. 323-336.*

[25] Valiant L.G., *Short monotone formulae for the majority function. Journal of Algorithms, Vol. 5 (1984), pp. 363-366.*

[26] Van Leijenhorst D.C., *A note on the formula size of the "mod k" functions. Info. Proc. Letters, Vol. 24 (1987) pp. 223-224.*

[27] Wallace C.S., *A suggestion for a fast multiplier. IEEE Trans. Electronic Comp. EC-13 (1964) pp. 14-17.*

[28] Zwick U., *A 4n lower bound on the combinational complexity of certain symmetric Boolean functions over the basis of unate dyadic Boolean functions.* To appear in *SIAM J. Comput.*